Readings Averaging

Accelerator protocol feature for local stations

Sep 15, 1991

Introduction

The Linac operates at 15 Hz, and each cycle may or may not produce beam, according to the accelerator timing system clock events. For normal viewing of Linac device readings, as on a parameter page, it has been traditional for many years to display reading values averaged over beam pulses only, ignoring the pulses which have no beam. In the case that there are *no* beam pulses occurring in the averaging interval, then the average of readings for *all* cycles in the interval is shown. The averaging logic serves two purposes. The first is that data from beam pulses is selected preferentially over that from non-beam pulses. The second is that fluctuations are reduced in the displayed values.

The averaging logic has always been done by the application program that needs the values. In order to do it, the application must be aware of which cycles are beam cycles. This information is available at each Linac local station by a status bit that is wired to all stations. Note that this signal indicates that beam is *scheduled* to be delivered by the Linac; it does not *guarantee* that protons will in fact be accelerated. But it serves to provide a proper value to display beam-related data, such as beam current toroid readings. If a beam current reading were used in place of a status bit value to determine whether a pulse was a beam pulse, then it would not display correctly under conditions of low beam currents below the threshold value used for that determination.

The local station parameter page (on the small consoles) checks its own local beam status bit to decide whether the present cycle is a beam cycle. The Macintosh parameter page (written by Bob Peters) uses a pseudo-channel value from a specific local station that contains the same information. The Vax parameter page (written by Jim Smedinghoff) uses a beam toroid reading.

When the beam cycle status information is delivered over the network, rather than measured locally as in the local stations themselves, there is an additional problem of correlation with the present cycle's data. If the data to be averaged comes from a different source node than the beam status, one must insure that the beam status is known from the same cycle; otherwise, the average value may be diluted, especially for beam toroid readings. The Macintosh parameter page uses the data server for its requests, so the pseudo-channel reading is delivered in the same message along with the data to be averaged.

In case of the Vax parameter page, the data may not be deliverable in a single message until the entire Linac controls upgrade is in place, and all Linac data is requested from the Linac data server node. In the interim, while some Linac data comes via the PDP-11 front end and some comes from the server node or from a

especially true because the Vax data pool manager (DPM) does not provide support for 15 Hz correlated data, even when it *does* come from a single node; a requester only gets (via DPGET) the last value of a given data item that was received. Another data item is obtained by a separate call to DPGET, and there is no guarantee that another cycle's data message has not been received since the first call. In fairness, it should be pointed out that one can obtain a sequence number for each item to determine whether new data has been received, but there is no way to insure that one can capture correlated data, even when the network actually delivers it. For Linac studies, this is a serious limitation of the accelerator control system, in the opinion of this writer.

In part because of the difficulties mentioned above, people have several times requested that the local stations perform the averaging logic. I have resisted this suggestion in the past because it seems to me that it amounts to placing application-specific code in the local station. One does not want to reach a point where applications can only be written on the Vax by adding application-dependent code to the local station system software as well. The local station's main job is to deliver the data to any and all requesters; what a requester *does* with that data should be up to the application program completely. In this case, however, the reality of the situation is that the Vax software is not designed to retrieve correlated 15 Hz data. If support for average data readings is to be provided to Vax applications, such as the parameter page, it may *have* to be done by the local station software.

Implementation

Although the averaging logic is well-known, it does not easily fit into local station support for data requests. The idea of averaging all data readings in the local station is rejected *a priori*; therefore, the support should be provided on a request basis. How does one specify to the local station that average values are needed? One suggestion is that a request period which is neither one-shot nor 15 Hz would be the indicator that average values are desired. The notion here is that sampling data from a 15 Hz Linac at other than 15 Hz is at best a hit-or-miss proposition and not for serious data-taking. Of course, given the averaging logic as stated above, the result of "averaging" data readings from a single pulse must be the same as the single reading from that pulse, independent of whether it was a beam pulse. This means that all readings could be treated with the same logic without regard for the data request period.

For the accelerator data request protocol supported by the RETDAT network task logic, which is the protocol used by the Vax consoles, each device request packet included in a data request message contains an SSDN. Inside the SSDN is a listype#, the fundamental data type specification used for local station data requests. In order to provide reading values useful for plotting as well as as for

any data request packet. Also, the check could be made for the reading property index where two bytes of data are requested. Note that we only want to average analog readings. There is a potential problem with using the readings listype# as a key. We will be providing reading words as basic status values that contain collections of status bits. This will be done to provide a name and alarm mask for such assembled status words. We should therefore include the check on the reading property.

Internally, the averaging requires additional storage for the accumulation of reading values over the beam cycles (or the non-beam cycles in the absence of beam cycles) for each device for which a reading is requested.

The request period is used to specify the averaging interval. This interval is not synchronized with anything but 15 Hz cycles. For the Linac which often delivers 13 successive pulses of beam to the Booster, the averaging interval may not include the entire sequence of 13. This points out one advantage of doing the averaging logic in an application, where an adjustment can be made to synchronize the averaging interval to such bursts of beam pulses for display purposes. A periodic request does not specify this kind of “breathing” logic. If it were done, a Vax program might not mind, but the server node might have a problem adjusting to it, since the server node delivers replies to requests at times which depend only on the fixed request period intervals. On the other hand, since the server is not doing the averaging, its last data readings received from the contributing nodes will already be averages. If the server node does not mind the “breathing” in the timing of the contributing nodes, it might be ok. (The server node is not given the job of doing the averaging because it is already a bottleneck by design, and it would require collecting the data from the contrib uting nodes at 15 Hz, while only delivering 1 Hz replies to the consoles. If the averaging is done locally by each contributing node, then the load is distributed; and the load is also much less, because those nodes only send their average values at 1 Hz to the server.) Another value in having the application program support averaging is that the count of the number of beam pulses present in the accumulation of the average can be shown on the display as well. This was done in the olden days for the Linac-only parameter page.

Details

Concentrate on the logic involved in support for the non-server request using the accelerator protocol. (All accelerator protocol logic is in the ACREQS module.) One memory block used in support of these requests is the type#14 internal ptrs block. Each device in a data request is “compiled” into an internal ptr, which is used to facilitate update of the reply data. Most often, the value of an internal ptr is the memory address of the data to be returned. Specifically, in the case of a request for an analog reading, the internal ptr is the address of the reading data

To support averaging, we still want to maintain a ptr to the reading field, but we must also maintain a longword accumulation value. The allocation of memory needed for the internal ptrs block depends upon the number of internal ptrs needed. We must add an extra longword for each internal ptr that needs to support averaging. During the scan in the NSERVER routine, we should detect the need for this extra space for the internal ptrs block and increase the value of NPTOTAL accordingly, since it is used later to allocate the internal ptrs block. The space for the averaging case will be two longwords per device; the first will be the usual internal ptr value, and the second will be the accumulation longword.

In addition, for the entire request, we need to keep two counters. One is FTDC, the count of cycles over the request period, and the other is SUMC, the number of cycles of data that have been added to produce the accumulation. Also, there must be a state bit that records whether the accumulation holds data from beam cycles or non-beam cycles.

Until now, a call to ACUPDATE from ACUPDCHK produces a new reply to a data request. To support this new feature, we must do some accumulation work *every* cycle, not just the cycles on which a reply is due. A test for the need of accum ulation logic is required in ACUPDCHK before ACUPDATE is called.

During initialization of the non-server request, each device request block (DRB) which needs the special averaging treatment is marked by setting the sign bit of the RDI word in the DRB. (The RDI word contains the read-type routine index and is a small integer used in the READTYPE call to update the answers corres ponding to that DRB.) The test for the need of the averaging logic is that the reading property index (=12 decimal) is used, the listype for analog reading (=0) is used, and the #bytes requested is 2. The internal ptr for this analog reading case will of course be the pointer to the reading word in the analog channel ADATA entry, so the accum ulation logic is simple. If other listypes should need averaging support in the future, they might be additionally permitted. As stated before, the number of internal ptrs required should double for such DRB's to allow room in the internal ptrs block allocated later for the longword needed for the accumulations. Set a flag so that the internal ptrs block header can be marked to indicate that averaging logic is needed by at least one DRB in the request.

For each cycle of a request that needs the averaging logic applied, according to this flag in the internal ptrs block header, the DRB's are scanned in AVGACCUM. For each DRB marked in its RDI word, accum u lations of the current readings are made according to the averaging algorithm.

Later, during the update scan at the end of the periodic request interval, the sign bit of the RDI word is tested to direct the updating loop to calculate and return

data values accumulated to obtain the average.

Postscript

This document was a working document used to explore an implementation that would accomplish this averaging feature for the local stations. Most of it was written before the code was started, but it is now updated to reflect what was done to implement the feature. The implementation required 140 lines of source code, about 350 bytes of object code and two days of coding and documen tation effort beyond the initial design discussions and contemplation.